Information System Prototyping*

Felix Voigtländer

RWTH Aachen, Aachen 52062, Germany

Abstract. Traditional development projects are rigid and do not involve customers and users effectively during development, resulting in not met requirements and unsatisfied customers. This paper explains prototypes and prototyping, and tries to reduce the risks of their use in a project by using an agile approach to software development while also giving an example.

Keywords: Prototyping · Rapid Prototyping · Agile · Kanban · User Interface

1 Introduction

Traditional software methodologies have many issues such as unsatisfied customers, minimal and late user involvement during development and long development cycles resulting in high development cost and time or even failed projects.

Prototyping is a tool that addresses these issues and has been effective and efficient in software development, especially in the context of user interface design.

This paper addresses the issues of conventional methodologies, explains prototypes and prototyping, discusses the advantages and disadvantages of the use of prototyping. The issue of the difficulty in managing projects using the prototyping approach is addressed with the agile methodology and its models of which the Kanban model is explained in greater detail, then an example is given and finally summarizes the use of prototyping and rapid prototyping.

2 Issues of traditional methodologies

Traditional methodologies such as the waterfall model [7], which will be explained later on, are simple to understand approaches to software development, which makes it easy to manage. All stages are clearly defined and a detailed design document reduces the risk of feature creep and increased development time and cost. Due to the static general terms and conditions set at the beginning of the Project production can be efficient and predictable while reducing waste, unnecessary features, and content.

* Supported by RWTH Aachen

The waterfall model is one of the simplest and popular models in the software development paradigm. The model is based on a sequential and linear design process, meaning each phase only begins once the previous phase has been completed. The waterfall model consists of five consecutive phases called requirements, design, implementation, and maintenance. As the names of the phases already suggest, actual working executable software is only produced very late in development, reducing the chance of finding problems with design and changing requirements early in development.

These models do not offer flexibility to changing requirements and due to the tendency of traditional models for long development cycles user feedback can only be gathered at the end of projects [2] increasing the time and cost of further alterations. Especially when complex factors are involved, such as human-machine interaction, it becomes problematic to make predictions in conventional methodologies [6]. The mutual understanding of the vision of a project is also not ensured in traditional approaches, since as [6] puts it "Clients don't know their requirements until the see them implemented"

Therefore conventional methodologies do not offer satisfactory results [6]. These approaches do not reduce development time and cost, yield unsatisfied costumers, even with extensive documentation communication problems are not reduced and there is no guarantee that the right product is produced [6].

3 Prototypes

A prototype is "a working model of (parts of) an information system, which emphasizes specific aspects of that system" [1]. These Prototypes are unfinished, but represent the vision of the final product and may vary depending on the state and needs of a project. Prototypes may evolve through iteration into the final product or throwaway prototypes may be discarded after their use. Low-technology prototypes such as the use of post-it notes or flip-charts may also be used, which empower the user to alter prototypes without any kind of technical-knowledge [1].

Prototypes may be categorized into horizontal and vertical prototypes, in which horizontal prototypes summarize the incomplete development of most system functions, while vertical prototypes include selected complete features or system functions. Incremental prototypes involve a series of horizontal and vertical moves [1].

Prototypes may also be distinguished via their functionality. Visual prototypes display the "look and feel" of the final product, but might have little to no functionality [5]. These are often used at the beginning of a project and to iterate user interfaces and are used as a reference for the final product, but are ultimately thrown away. On the other hand, we have executable prototypes, which are useable and often evolve into the final product. Executable prototypes may also be used to explore new functionalities and to test their feasibility[5].

Another form of classification for prototypes are Floyd's 3"E"s. Exploratory prototypes are used to evaluate requirements and their potential solution. Such prototypes can be functional and visual prototypes. Experimental prototypes involve the implementation of selected requirements and features. Their evaluation result in the feasibility and suitability of the implemented features. Lastly, evolutionary prototypes evolve into the final product and adapt to changing requirements. Such prototypes may also be used for pilot systems [3].

In a nutshell, many kinds of prototypes can be used to fit the needs of the project at hand.

4 Prototyping

Prototyping is the simple procedure of using prototypes during the project, which should be combined with others[1]. During prototyping models of the product are shown to the user or customer, which are then discussed to evaluate problems and further advancements. Prototyping can be used when requirements are unclear especially with systems with high user interaction or systems require a higher form of cognitive processing from users such as "soft-skills". The use of prototypes also helps the communication between everyone involved in the project, serving as a reference for everyone involved [5].

The use of prototyping may require media which makes prototyping efficient and effective. Such tools need to offer modularity and plasticity. Modularity enables the developer to add, remove, and alter modules, without causing many issues in other systems. Plasticity allows alterations without significant time and cost penalties [6]. The Rapid synthesis of prototypes is also useful with the use of prototypes[6]. To ensure stability throughout many iterations during the project a layered approach may be used as well [2].

A special form or prototyping is rapid prototyping (RP). RP acknowledges the complexity of aspects in development, rather than trying to minimize the complexity. This methodology expects that the first approach is probably faulty, but iterates and improves the product with the discovery of problems. The goal of RP is to minimize the issues related to traditional development approaches like the waterfall model [6].

5 Current State

A study by the Philipps-University Marburg [2] used an application framework and rapid application development approach for their information system in healthcare. Lenz and Kuhn[2] explain that traditional information system development approaches, like the waterfall model, are inadequate for rapid application development (RAD) since they do not consider requirement changes during development. Traditional software engineering methods also tend to long development cycles, thus late feedback from users.

To apply rapid prototyping to huge healthcare information systems while maintaining flexibility and stability a layered approach to system evolution with the addition of a generator-tool for application development have been used.

The paper describes the disadvantages of heterogeneous databases and modules, due to module integration from different developers, and bugs in lower layers causing complex and hard to debug issues in higher layers, for which smaller and robust core systems for future systems are proposed. Yet, prototyping yielded satisfactory results with high quality.

Another study by Jones and Richey [5] investigated the use of rapid prototyping for two projects in instructional design by designers and customers.

The projects showed the development of two products of high quality in a short period. The customer was involved in development early on and was satisfied with its results. Tasks could be completed concurrently and collaboratively, which is not possible with the linear approach of conventional methods.

The paper concluded, that projects created using the Rapid Prototyping methodologies created useable products, that were delivered in a shorter time, and were received by satisfied customers. Rapid Prototyping appears to provide one solution to produce high-quality products in less time.

In a nutshell, both papers concluded that the use of prototyping yielded satisfying results with high quality, with low development time and costs, if the known issues of prototyping have been addressed correctly.

6 Benefits and Problems with Prototyping

Prototyping as well as Rapid Prototyping aim to reduce the issues related to conventional development methodologies. Prototyping benefits the communication and solves communication problems across the team, customers and users[6] serving as a reference point already early and during development[1]. The customer is left satisfied, by being able to be involved during the development and can help with feedback to produce a better product. Also a tendency of customers and users buying into prototypes result in a greater commitment in a project. When discussing prototypes, requirements are better outlined from all parties involved causing the reduction of unnecessary requirements and features, cutting cost and development time [1]. With adjusted requirements, late alterations with high development times and costs are avoided and found early in development instead.

Yet, continual user participation and its rapidity may become stressful for developers, which might need additional skills and knowledge about prototyping tools. Early user involvement might oversell the final product leading to the creation of unrealistic requirements, making it harder to manage user expectations. Large systems are difficult to prototype since the complete infrastructure needs to be set up first[1].

Projects using prototyping become harder to manage. The cost of development effort is increased due to the development of throwaway prototypes and creeping scopes, resulting in hard to budget projects and longer development cycles.The use of prototypes may also result in a design-by-repair approach and may reduce creativity because of developers buying into one prototype instead of finding a better solution.

7 The Agile Methodology

To address the issues with the management of a project using the prototyping approach different methodologies other than conventional methodologies, like the waterfall model, need to be used. A promising approach is the "agile" approach to project management. It can be used in combination with prototyping.

The most important thing to note is that agile is not defined by tools, models, or a framework, but rather by a mindset[8]. This mindset consists of its values and principles, which helps on how to approach a project, it relies on the collaboration in teams and the flow of information. Agile is also customer-centric and puts him as part of the development team. Yet to give the mindset a structure and to ease the flow of information agile practices and models may be used.

This mindset is set by the Agile Manifesto[8], which specifies its values:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Such values address the issues coming from prototyping. It emphasizes the collaboration of teams and customers, working software and responding to change, while ignoring the use of processes, tools, documentation, contract negotiation and following a plan.

Team collaboration is at the forefront of agile development, thus it is about communication and working together. Finding problems together and solving problems together. This is valued more than using tools and processes, which are be rigid and outdated and thus cannot react effective and efficient enough to a changing development environment.

Producing working software and prototypes to gain direct feedback from customers or consumers to finding problems early on, which might have been unknown during the planning stages of development. In case of issues alterations are less costly to implement, since less systems rely on newly implemented systems. This is valued more than comprehensive documentation, which might become outdated, imprecise and faulty. With the value being set, systems can be structured accordingly, as example in [2], to address issues of instability from many iterations.

With the use of early working prototypes, the customer can give direct feedback early on, simplifying changes which are easy to adapt, leading to greater customer satisfaction and more predictable software development timelines. This is valued more than extensive contract negotiations with the customers at the start of the project, which become outdated to which the customer is stuck and might be unsatisfied with or result in revisions late in development with higher development cost and time.

It is more efficient and cost-effective to respond to change and to adapt instead of following an outdated plan, which might even be faulty.

To summarize, the agile methodology is a mindset for software development. It is outlined by its values[8] which should be seen as guidelines, rather than rules to follow.

To give the mindset of the agile approach to software development a structure and to make it easier to use, agile methods might be applied. These incorporate the values and principles of agile software development. Each method can be adjusted to fit the needs of the team or the project to be as effective as possible. Further changes to models can be implemented at any time to improve it. Here are some examples of agile methods:

1. SCRUM is a framework which uses an iterative, incremental approach, which is self-improving through reviews and self-reflection. Thus it is flexible and adaptive and aims to improve the development cycle by periodic reevaluations. This makes it also more effective and gradually increases quality.
2. Extreme Programming (XP) is for changing or unknown requirements in a project for small to medium-sized teams. It also consists of lots of testing and pair programming.
3. Kanban is a visual management method separated into different sections for parallel tasks with a task limit per section. The model shows bottlenecks and is self-improving. This method will be elaborated in the next section.

8 Kanban Methodology

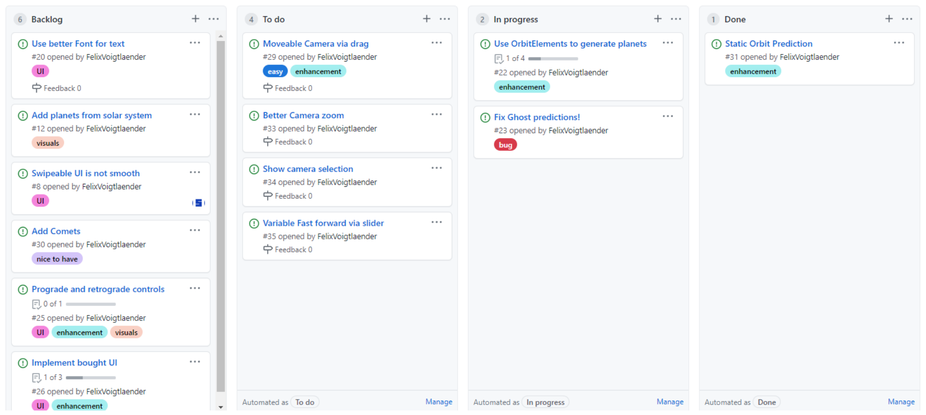
Kanban was developed by Taiichi Ohno in 1940 for Toyota in Japan [10]. It aimed to be a simple framework for managing tasks and inventory for every stage in automotive production. The goal of this model was to improve productivity compared to American rivals and to avoid bottlenecks and over-stocking during production.

Kanban is a tool for visualizing workflow and managing it. It limits work in progress, improves the flow of parallel tasks in its development and is a great tool for teams to self manage. This model does not force organizational, structural, or process changes, instead change happens iteratively and slowly for the team to adapt to. The main goal of this model is to improve the flow of information and to ensure a steady workflow without bottlenecks. Thus the board should be easy to read and understandable.

For visualization, a board with successive sections and tasks are used. The board can be a physical board with post-it notes or an electric board like Trello. The Tasks move from one section to another according to their state. These Tasks may be features, changes or even bug fixes, which should be manageable by one person. Sections may have different policies, for example a definition of done, thus which criteria need to be met for a task to move on. All Sections have a maximum number of tasks which they can hold, once a section reaches its maximum number of tasks no tasks are allowed to be moved to this section. The section must be freed up by completing its tasks and moving them on. Each iteration can be a week or a month, with the preference for smaller timescales.

In each iteration tasks may be selected, which should be completed by the end of the iteration. It is advantageous to select tasks, which result together in working software or prototypes.

A basic structure of this model, see figure 1, consists of the following sections, which will be explained in detail: backlog, todo, in progress, and done. Naming, the number of sections and structure may be changed to better suit the team.


6 Backlog

Use better Font for text
#20 opened by FelixVoigtlaender
UI
Feedback 0

Add planets from solar system
#12 opened by FelixVoigtlaender
visuals

Swipeable UI is not smooth
#8 opened by FelixVoigtlaender
UI

Add Comets
#30 opened by FelixVoigtlaender
nice to have

Prograde and retrograde controls
0 of 1
#25 opened by FelixVoigtlaender
UI enhancement visuals

Implement bought UI
1 of 3
#26 opened by FelixVoigtlaender
UI enhancement

4 To do

Moveable Camera via drag
#29 opened by FelixVoigtlaender
easy enhancement
Feedback 0

Better Camera zoom
#33 opened by FelixVoigtlaender
Feedback 0

Show camera selection
#34 opened by FelixVoigtlaender
Feedback 0

Variable Fast forward via slider
#35 opened by FelixVoigtlaender
Feedback 0

Automated as To do Manage

2 In progress

Use OrbitElements to generate planets
1 of 4
#22 opened by FelixVoigtlaender
enhancement

Fix Ghost predictions!
#23 opened by FelixVoigtlaender
bug

Automated as In progress Manage

1 Done

Static Orbit Prediction
#31 opened by FelixVoigtlaender
enhancement

Automated as Done Manage

Fig. 1: Example Kanban Board [12]

The “Backlog” holds all tasks for the project which need to be done for the final product. New tasks may be added for new features, bugs or changes. In each iteration tasks may be selected from the backlog and moved to the “todo” section.

The “todo” section holds all tasks which need to be completed and moved to the “done” section by the end of the current iteration. Thus tasks need to be selected such that the team is not over nor under burdened. New tasks shouldn’t be added from the backlog during an iteration, to keep the flow manageable and to avoid further issues. Once a task is being worked on it is moved to the “in progress” section.

The “in progress” section holds all tasks, which are being currently worked on. This section may also hold a review section to which tasks are moved once they are completed and to be tested and reviewed. Finally the completed tasks are moved to the “done” section.

The “done” section holds all tasks which are completed.

In this example at the beginning of each iteration tasks are selected from the “backlog” section and moved to the “todo” section, these tasks should be completed by the end of this iteration. Developers can now pick their tasks and move them to the “in progress” section and work on them. Once the task is completed the developer can move their task to the “done” section, in case of a review section the developer moves the task to the “review” section instead, which will then be reviewed. In case of issues the task might be moved back to the “todo” section with a description of an issue or a bug report may be formulated.

In case the “in progress” section reaches its maximum number of tasks, new tasks from the todo list are not allowed to be moved to the “in progress” section. Instead, developers must complete their tasks before starting new ones and freed up developers can help others.

The agile Kanban model is simple to understand and easy to manage. It visualizes the workflow clearly on a board, which is straightforward to read for people of different backgrounds. It enables the team to self manage task allocation decreasing the number of bottlenecks during development. The model is flexible and makes it easy to adapt to changes during development. It is consistent and easily measurable and always improvable, by altering structures, definitions, or tasks. Changes are gradual, making it easier for a team to adapt.

Yet clustered boards, with too many tasks or even outdated ones can lead to issues in development. Overcomplicating the model may also lead to slower development and communication. Overestimating or underestimating the workload, thus selecting the wrong amount and types of tasks in each iteration leads to slower development and other issues like wasted resources and a clogged up pipeline.

In a nutshell, the Kanban model is a useful tool for workflow visualization and management. And if used correctly, thus the board is simple to read with correct iteration estimates, the board can be an effective tool for efficiency and self-improvement. The model is also very adaptable since at first only the board and iteration plannings need to be introduced, without having to change large portions of team structures.

9 An Example

In a student project[11] rapid prototyping has been used during development and for the creation of its user interface. The project aims to create an orbit based puzzle game on mobile devices with high influence from the desktop game Kerbal Space Programm. For the static orbit prediction of planets Keplerian elements and his laws of planetary motion are used. The dynamic predictions for the user-controlled spaceship use a performance adapted gravity system approach and simple Newtonian calculations. The project has been in development for 3 months and is aimed to release in 6 months.

For modularity, plasticity and rapid synthesis of prototypes the engine Unity has been used, which uses a component-based system and as scripting language C# is used. In combination with rapid prototyping an altered version of the agile Kanban[10] model is used to ensure manageability throughout the project, see figure 1. Such alterations include setting the definition of sprints as the completion of tasks in the ”Todo” section, instead of set time cycles of one week. Once all tasks of a sprint are completed a new prototype with the changes is distributed to users. Additionally, once tasks start clogging up the workflow, those tasks are reevaluated and might be decomposed into new smaller tasks. To reduce the risk of feature creeping three design pillars have been defined that need to be addressed and may not be contradicted by additional features or content. These design pillars are minimalism, education and gratification. Users are able to download the latest prototypes/builds from Google Drive[13] and sessions in which the user is observed while using the prototype are done.

This paper addresses the prototyping approach for developing the user interface during puzzle sequences. The input options include the setting of thrust direction, thrust ignition, pausing and unpause, different time scale options, saving and loading, zoom and target of the camera, and leaving the puzzle sequence. The interface displays the solar system, the current predicted orbit of spaceship and planets, thrust stages left, and the estimated prediction if the thrust is ignited.

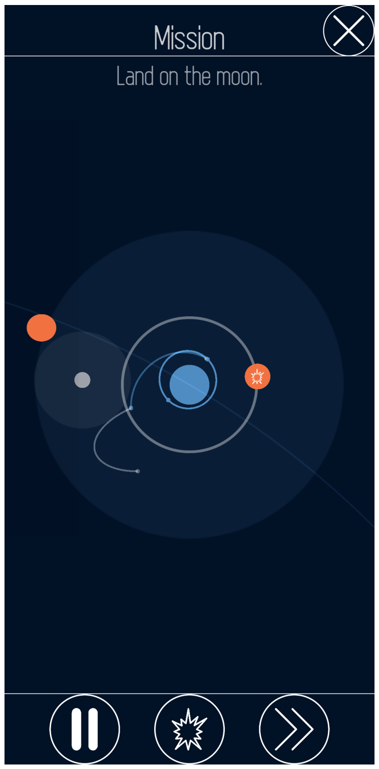


Fig. 2: Prototoype 1 [14]

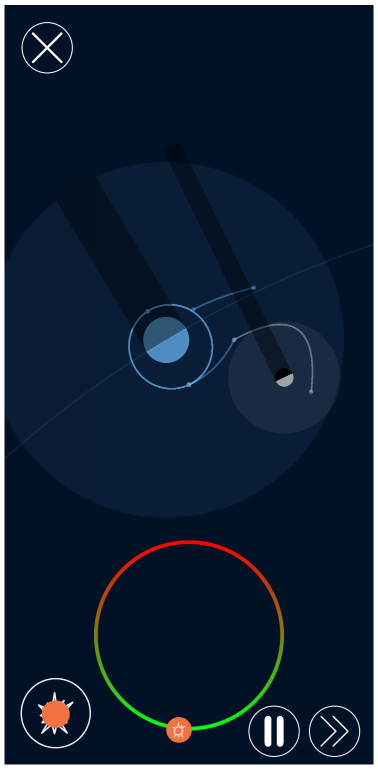


Fig. 3: Prototoype 2 [15]

In the first initial prototype, see figure 2, the joystick for thrust direction is located at the center, a button for thrust ignition is positioned at the bottom center next to pause/unpause and fast forward. Thrust stages are positioned at the middle left.

The initial feedback by users was very helpful. It was pointed out that the user interface seems very cluttered, the orbit paths were hard to see and buttons did not show if the system was paused or not. During observation of users interacting with the user interface more problems were discovered. At first most users clicked on the staging, probably to ignite the stage, instead of the designated ignite button. Additionally some users accidentally clicked the button to close the scenario. Also the joystick for thrust direction seemed to confuse some participants, especially during the simulation.

The second prototype, see figure 3, tries to address most issues. Unneeded lines and information has been removed or moved to the bottom or top of the screen, to declutter the user interface. The joystick for the thrust direction has been moved to the bottom center of the screen, with the addition to being relative to the velocity vector of the spaceship. Also staging and the ignition button have been moved to the same location, as to show which stage is ignited once pressed. The close button has been moved from right to left to address the issue of users accidentally closing the scenario.

Prototyping has been useful for the design of the user interface. Completely unknown issues, such as the staging and ignition button dependency, have been discovered early in development and could be fixed without cost and time penalties, which would have been caused in later stages of development. Additional feedback and observations from the second prototype have already been discovered. Such are the issue of multiple types of time scales, which will be addressed via a slider for pause, normal time and several types of time scales. And additional issues with other user interface panels, relating to this user interface panel.

Finally, prototyping has been helpful in finding and addressing issues found in user interface elements and system functions. This approach will be used for further development.

10 Discussion

We presented the benefits and problems of prototyping in information systems and included approaches to mitigate the problems of this approach. This section is going to discuss how well the approaches deal with the problems.

The early involvement of users oversell the final product with the creation of unrealistic requirements. The importance of educating the user has been discussed in the shown examples [2, 1], also the understandable Kanban board lets the user be involved in the development and understand the development hindering the user in Overestimating the possibilities of the final product.

Large systems are hard to prototype since multiple systems have to be set up to be able to test the whole system. In the example of the development of an information system in healthcare [2] a layered approach has increased the effectiveness of prototyping by being able to prototype in higher layers with the help of generator tools. Also, in our example we used a platform supporting rapid prototyping and a component-based approach to be able to prototype more frequently and in a stable manner.

The difficulty of managing a project using prototyping is improved by using an agile approach. The complexity and the fast-paced iteration cycles is expected and reacted to in accordance to the agile principles. The agile methodologies do give a structure and measurable environment as a tool for a more effective and efficient development.

To conclude, prototyping poses known risks, which are expected and can be mitigated by approaching development with an agile mindset and an agile framework which can be adjusted during a project to react more effective to the needs and problems of the project.

11 Conclusion

Prototyping and Rapid Prototyping as tools for development can be useful to address the issues found in conventional methodologies if used correctly. It addresses the long development cycles, high development costs and assures customer and user satisfaction, due to their involvement early and during development. Complex situations can be explored inexpensively with prototypes and iteratively improved.

Issues with prototyping need to be taken in consideration. Large systems are hard to prototype, user expectation needs to be managed and the management of projects using this approach is difficult.

But, in consideration of problems which might arise from prototyping and its advantages prototyping is a powerful tool for software and user interface development.

References

1. Paul Beynon-Davies, Douglas Tudhope, Hugh Mackay: Information systems prototyping in practice. Journal of Information Technology **14**(1), 107–120 (1999)
2. R. Lenz, K.A. Kuhn: Towards a continuous evolution and adaptation of information systems in healthcare. International Journal of Medical Informatics **2004**(73), 75–89
3. Dirk Bäumer, Walter R. Bischofberger, Horst Lichter, Heinz Züllighoven: User Interface Prototyping - Concepts, Tools, and Experience. Proceedings - International Conference on Software Engineering 1996, DBLP https://doi.org/10.1109/ICSE.1996.493447
4. Soukaina Merzouk, Abdessamad Cherkaoui, Abdelaziz Marzak, Saek Nawal: IoT methodologies: comparative study International Workshop on Artificial Intelligence and Internet of Things (A2IoT) August 9-12, 2020, Leuven, Belgium
5. Toni Stokes Jones, Rita C. Richey: Rapid Prototyping Methodology in Action: A Developmental Study ETR&D, Vol.48, No. 2, 2000, pp. 63-80 ISSN 1042-1629
6. Steven D Tripp, Barbara Bichelmeyer: Rapid Prototyping: An Alternative Instructional Design Strategy ETR&D Vol. 38, No. 1, pp. 31-44 ISSN 1042 1042-1629

7. Dr. Winston W. Rovce: MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS (1987)
8. The Agile Manifesto, https://agilemanifesto.org/. Last accessed 15 Nov 2020
9. Feature Creep, https://www.agile-academy.com/de/agiles-lexikon/feature-creep/. Last accessed 15 Nov 2020
10. Eric Brechner: Agile Project Management with Kanban. Microsoft Press, Redmond, Washington 98052-6399 (2015)
11. Student Project: Github repository of the project, https://github.com/FelixVoigtlaender/Planeter---Mini-Ksp Last accessed 14 Dec 2020
12. Kanban board: example board from student project, https://github.com/FelixVoigtlaender/Planeter---Mini-Ksp/projects/1 Last accessed 13 Jan 2021
13. Student Project: Collection of all prototypes, https://drive.google.com/drive/folders/1-CI-kAiKsY4FEDwQmcAWbDM9oTFoPtvh?usp=sharing Last accessed 14 Dec 2020
14. Prototype 1: 2020-11-08_Build4.apk , https://drive.google.com/file/d/1pbc-fyS5lXyyZoWz7Kb9KhB5P1FEOhCf/view?usp=sharing Last accessed 13 Jan 2021
15. Prototype 2: 2020-11-13_Build1.apk , https://drive.google.com/file/d/1pAZzR77ZeYCSorazqdkINXD2tKCASWBk/view?usp=sharing Last accessed 13 Jan 2021